

# **Отчет по лабораторной работе №4**

**Дисциплина архитектура компьютера**

Паласиос Фелипе

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>9</b>
<b>4</b>	<b>Выводы</b>	<b>17</b>
	<b>Список литературы</b>	<b>18</b>

## Список иллюстраций

3.1	текстовый файл с именем hello.asm . . . . .	9
3.2	файл . . . . .	10
3.3	Транслятор NASM для компиляции приведённого выше текста программы «Hello World» . . . . .	11
3.4	Созданные файлы . . . . .	12
3.5	help . . . . .	13
3.6	выполнение . . . . .	14
3.7	выполнение . . . . .	14
3.8	файлы hello.asm и lab4.asm . . . . .	15
3.9	Github . . . . .	16

## Список таблиц

# 1 Цель работы

Целью работы является освоение процедуры оформления отчетов с помощью легковесного языка разметки Markdown.

## 2 Задание

Порядок выполнения лабораторной работы

Программа Hello world!

1. Рассмотрим пример простой программы на языке ассемблера NASM. Традиционно первая программа выводит приветственное сообщение Hello world! на экран. Создайте каталог для работы с программами на языке ассемблера NASM:

```
mkdir -p ~/work/arch-pc/lab04
```

2. Перейдите в созданный каталог

```
cd ~/work/arch-pc/lab04
```

3. Создайте текстовый файл с именем hello.asm

```
touch hello.asm
```

4. откройте этот файл с помощью любого текстового редактора, например, `gedit gedit hello.asm` и введите в него следующий текст:

5. Транслятор NASM для компиляции приведённого выше текста программы «Hello World» необходимо написать: `nasm -f elf hello.asm` Если текст программы набран без ошибок, то транслятор преобразует текст программы из файла hello.asm в объектный код, который запишется в файл hello.o. Таким образом,

6. Расширенный синтаксис командной строки NASM Полный вариант командной строки `nasm` выглядит следующим образом: `nasm [-@ косвенный_файл_настроек] [-o объектный_файл] [-f формат_объектного_файла] [-l листинг] [параметры...] [-] исходный_файл` ✘ Выполните следующую команду: `nasm -o obj.o -f elf -g -l list.lst hello.asm` Данная команда скомпилирует исходный файл `hello.asm` в `obj.o` (опция `-o` позволяет задать имя объектного файла, в данном случае `obj.o`), при этом формат выходного файла будет `elf`, и в него будут включены символы для отладки (опция `-g`), кроме того, будет создан файл листинга `list.lst` (опция `-l`). С помощью команды `ls` проверьте, что файлы были созданы.

7. Компоновщик LD чтобы получить исполняемую программу, объектный файл необходимо передать на обработку компоновщику: `ld -m elf_i386 hello.o -o hello`

С помощью команды `ls` проверьте, что исполняемый файл `hello` был создан. Компоновщик `ld` не предполагает по умолчанию расширений для файлов, но принято использовать следующие расширения: • `o` – для объектных файлов; • без расширения – для исполняемых файлов; • `tar` – для файлов схемы программы; • `lib` – для библиотек. Ключ `-o` с последующим значением задаёт в данном случае имя создаваемого исполняемого файла. Выполните следующую команду:

```
ld -m elf_i386 obj.o -o main
```

Формат командной строки LD можно увидеть, набрав `ld -help`.

8. Запуск исполняемого файла Запустить на выполнение созданный исполняемый файл, находящийся в текущем каталоге, можно, набрав в командной строке: `./hello`

9. Задание для самостоятельной работы

10. В каталоге `~/work/arch-рс/lab04` с помощью команды `ср` создайте копию файла `hello.asm` с именем `lab4.asm`
11. С помощью любого текстового редактора внесите изменения в текст программы в файле `lab4.asm` так, чтобы вместо `Hello world!` на экран выводилась строка с вашими фамилией и именем.
12. Оттранслируйте полученный текст программы `lab4.asm` в объектный файл. Выполните компоновку объектного файла и запустите получившийся исполняемый файл.
13. Скопируйте файлы `hello.asm` и `lab4.asm` в Ваш локальный репозиторий в каталог `~/work/study/2023-2024/“Архитектура компьютера”/arch-рс/labs/lab04/`. Загрузите файлы на Github.



## 3 Выполнение лабораторной работы

Порядок выполнения лабораторной работы

Программа Hello world!

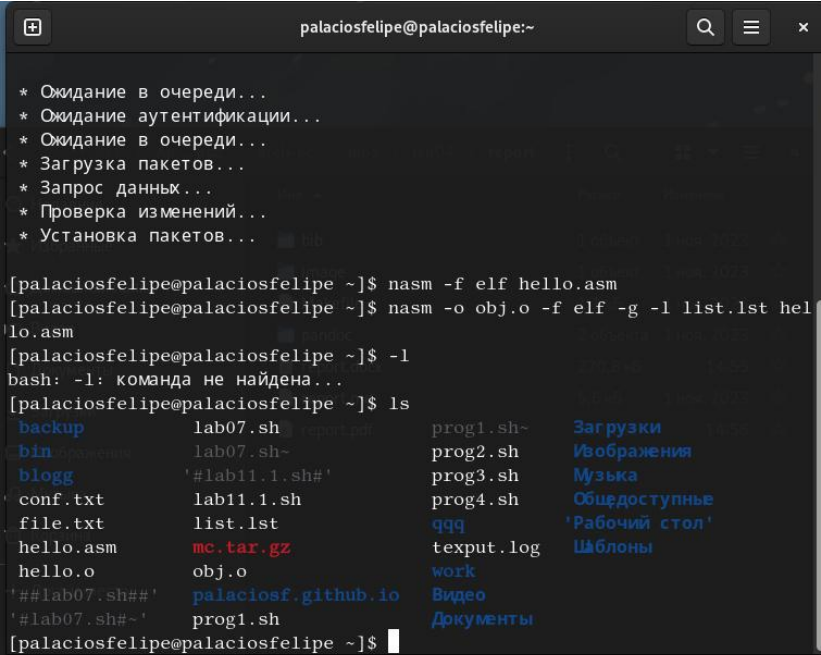
1. Создайте каталог для работы с программами на языке ассемблера NASM:

```
mkdir -p ~/work/arch-pc/lab04
```

2. Перейдите в созданный каталог

```
cd ~/work/arch-pc/lab04
```

3. Создайте текстовый файл с именем hello.asm (рис. 3.1).



```
palaciosfelipe@palaciosfelipe:~  
* Ожидание в очереди...  
* Ожидание аутентификации...  
* Ожидание в очереди...  
* Загрузка пакетов...  
* Запрос данных...  
* Проверка изменений...  
* Установка пакетов...  
[palaciosfelipe@palaciosfelipe ~]$ nasm -f elf hello.asm  
[palaciosfelipe@palaciosfelipe ~]$ nasm -o obj.o -f elf -g -l list.lst hel  
lo.asm  
[palaciosfelipe@palaciosfelipe ~]$ -l  
bash: -l: команда не найдена...  
[palaciosfelipe@palaciosfelipe ~]$ ls  
backup      lab07.sh      prog1.sh~    Загрузки  
bin         lab07.sh~     prog2.sh     Изображения  
blogg       '#lab11.1.sh#' prog3.sh     Музыка  
conf.txt    lab11.1.sh    prog4.sh     Общедоступные  
file.txt    list.lst      qqg          'Рабочий стол'  
hello.asm   mc.tar.gz     texput.log   Шаблоны  
hello.o     obj.o         work  
'##lab07.sh##' palaciosf.github.io Видео  
'#lab07.sh#~' prog1.sh      Документы  
[palaciosfelipe@palaciosfelipe ~]$
```

Рис. 3.1: текстовый файл с именем hello.asm

touch hello.asm

4. откройте этот файл с помощью любого текстового редактора, например, gedit (рис. 3.2).

gedit hello.asm



```
; hello.asm
SECTION .data ; Начало секции данных
hello: DB 'Hello world!',10 ; 'Hello world!' плюс ; символ перевода строки
helloLen: EQU $-hello ; Длина строки hello
SECTION .text ; Начало секции кода
GLOBAL _start
_start: ; Точка входа в программу
mov eax,4 ; Системный вызов для записи (sys_write)
mov ebx,1 ; Описатель файла '1' - стандартный вывод
mov ecx,hello ; Адрес строки hello в ecx
mov edx,helloLen ; Размер строки hello
int 80h ; Вызов ядра
mov eax,1 ; Системный вызов для выхода (sys_exit)
mov ebx,0 ; Выход с кодом возврата '0' (без ошибок)
int 80h ; Вызов ядра
```

Рис. 3.2: файл

и введите в него следующий текст:

5. Транслятор NASM для компиляции приведённого выше текста программы «Hello World» необходимо написать: `nasm -f elf hello.asm` Если текст программы набран без ошибок, то транслятор преобразует текст программы из файла `hello.asm` в объектный код, который запишется в файл `hello.o`. Таким образом, (рис. 3.3).

```
palaciosfelipe@palaciosfelipe:~$ nasm -f elf hello.asm
[palaciosfelipe@palaciosfelipe ~]$ nasm -o obj.o -f elf -g -l list.lst hel
lo.asm
[palaciosfelipe@palaciosfelipe ~]$ -l
bash: -l: команда не найдена...
[palaciosfelipe@palaciosfelipe ~]$ ls
backup      lab07.sh  prog1.sh~  Загрузки
bin         lab07.sh~  prog2.sh   Изображения
blogg      '#lab11.1.sh#'  prog3.sh   Музыка
conf.txt   lab11.1.sh  prog4.sh   Общедоступные
file.txt   list.lst   qqg        'Рабочий стол'
hello.asm  mc.tar.gz  texput.log  Шаблоны
hello.o    obj.o      work
'##lab07.sh##'  palaciosf.github.io  Видео
'#lab07.sh#~'   prog1.sh    Документы
[palaciosfelipe@palaciosfelipe ~]$
```

Рис. 3.3: Транслятор NASM для компиляции приведённого выше текста программы «Hello World»

## 6. Расширенный синтаксис командной строки NASM

Выполните следующую команду: `nasm -o obj.o -f elf -g -l list.lst hello.asm`  
Данная команда скомпилирует исходный файл `hello.asm` в `obj.o` (опция `-o` позволяет задать имя объектного файла, в данном случае `obj.o`), при этом формат выходного файла будет `elf`, и в него будут включены символы для отладки (опция `-g`), кроме того, будет создан файл листинга `list.lst` (опция `-l`). С помощью команды `ls` проверьте, что файлы были созданы. (рис. 3.4).

```
palaciosfelipe@palaciosfelipe:~$  
* Ожидание в очереди...  
* Ожидание аутентификации...  
* Ожидание в очереди...  
* Загрузка пакетов...  
* Запрос данных...  
* Проверка изменений...  
* Установка пакетов...  
[palaciosfelipe@palaciosfelipe ~]$ nasm -f elf hello.asm  
[palaciosfelipe@palaciosfelipe ~]$ nasm -o obj.o -f elf -g -l list.lst hel  
lo.asm  
[palaciosfelipe@palaciosfelipe ~]$ -l  
bash: -l: команда не найдена...  
[palaciosfelipe@palaciosfelipe ~]$ ls  
backup      lab07.sh      prog1.sh~    Загрузки  
bin         lab07.sh~     prog2.sh     Изображения  
blogg       '#lab11.1.sh#' prog3.sh     Музыка  
conf.txt    lab11.1.sh    prog4.sh     Общедоступные  
file.txt    list.lst      qqg          'Рабочий стол'  
hello.asm   mc.tar.gz     texput.log   Шаблоны  
hello.o     obj.o         work  
'##lab07.sh##' palaciosf.github.io Видео  
'#lab07.sh#~' prog1.sh      Документы  
[palaciosfelipe@palaciosfelipe ~]$
```

Рис. 3.4: Созданные файлы

7. Компоновщик LD чтобы получить исполняемую программу, объектный файл необходимо передать на обработку компоновщику: `ld -m elf_i386 hello.o -o hello`

С помощью команды `ls` проверьте, что исполняемый файл `hello` был создан. Выполните следующую команду:

```
ld -m elf_i386 obj.o -o main
```

Формат командной строки LD можно увидеть, набрав `ld -help` (рис. 3.5)

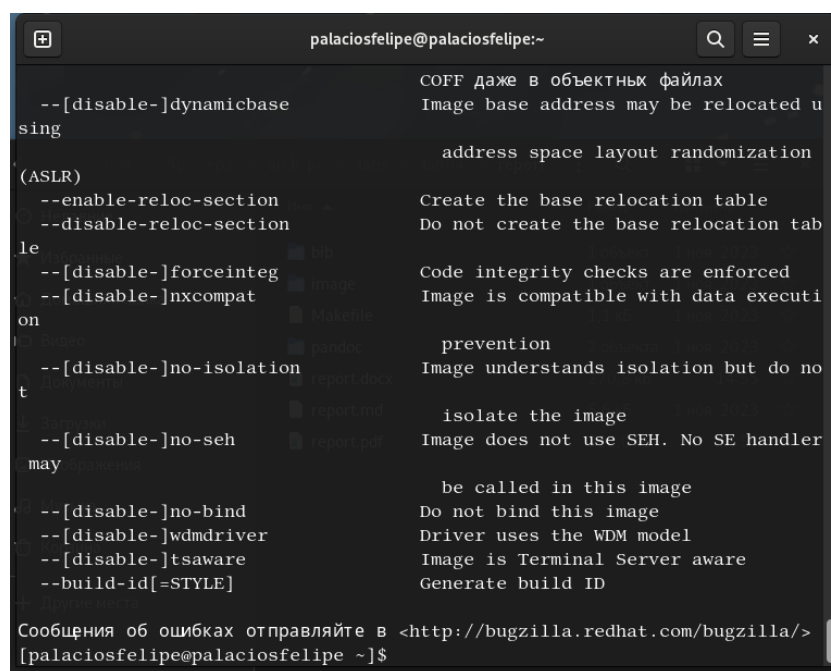
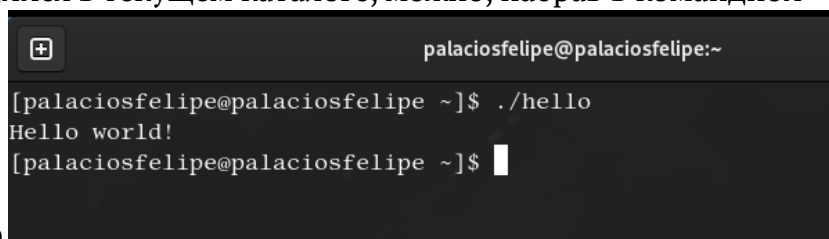


Рис. 3.5: help

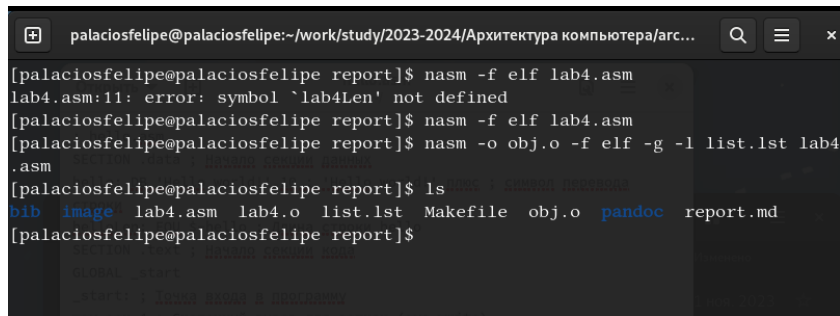
8. Запуск исполняемого файла Запустить на выполнение созданный исполняемый файл, находящийся в текущем каталоге, можно, набрав в командной



строке (рис. ??) ./hello

9. Задание для самостоятельной работы
10. В каталоге ~/work/arch-рс/lab04 с помощью команды `cp` создайте копию файла (рис. 3.6)

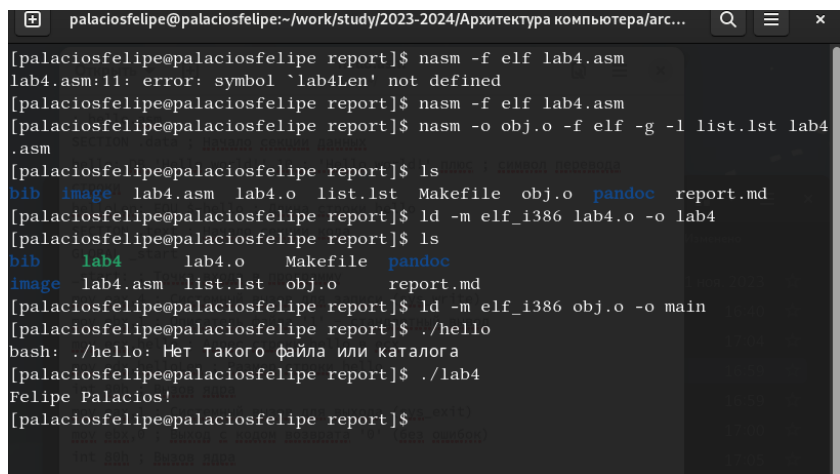
hello.asm с именем lab4.asm



```
palaciosfelipe@palaciosfelipe:~/work/study/2023-2024/Архитектура компьютера/arc...
[palaciosfelipe@palaciosfelipe report]$ nasm -f elf lab4.asm
lab4.asm:11: error: symbol `lab4Len' not defined
[palaciosfelipe@palaciosfelipe report]$ nasm -f elf lab4.asm
[palaciosfelipe@palaciosfelipe report]$ nasm -o obj.o -f elf -g -l list.lst lab4.asm
[palaciosfelipe@palaciosfelipe report]$ ls
bib image lab4.asm lab4.o list.lst Makefile obj.o pandoc report.md
[palaciosfelipe@palaciosfelipe report]$
```

Рис. 3.6: выполнение

11. С помощью любого текстового редактора внесите изменения в текст программы в файле lab4.asm так, чтобы вместо Hello world! на экран выводилась строка с вашими фамилией и именем
12. Оттранслируйте полученный текст программы lab4.asm в объектный файл. Выполните компоновку объектного файла и запустите получившийся исполняемый файл (рис. 3.7)



```
palaciosfelipe@palaciosfelipe:~/work/study/2023-2024/Архитектура компьютера/arc...
[palaciosfelipe@palaciosfelipe report]$ nasm -f elf lab4.asm
lab4.asm:11: error: symbol `lab4Len' not defined
[palaciosfelipe@palaciosfelipe report]$ nasm -f elf lab4.asm
[palaciosfelipe@palaciosfelipe report]$ nasm -o obj.o -f elf -g -l list.lst lab4.asm
[palaciosfelipe@palaciosfelipe report]$ ls
bib image lab4.asm lab4.o list.lst Makefile obj.o pandoc report.md
[palaciosfelipe@palaciosfelipe report]$ ld -m elf_i386 lab4.o -o lab4
[palaciosfelipe@palaciosfelipe report]$ ls
bib lab4 lab4.o Makefile pandoc
image lab4.asm list.lst obj.o report.md
[palaciosfelipe@palaciosfelipe report]$ ld -m elf_i386 obj.o -o main
[palaciosfelipe@palaciosfelipe report]$ ./hello
bash: ./hello: Нет такого файла или каталога
[palaciosfelipe@palaciosfelipe report]$ ./lab4
Felipe Palacios!
[palaciosfelipe@palaciosfelipe report]$
```

Рис. 3.7: выполнение

13. Скопируйте файлы hello.asm и lab4.asm в Ваш локальный репозиторий в каталог ~/work/study/2023-2024/“Архитектура компьютера”/arch-pc/labs/lab04/ (рис. ??) Загрузите файлы на Github (рис. 3.9)

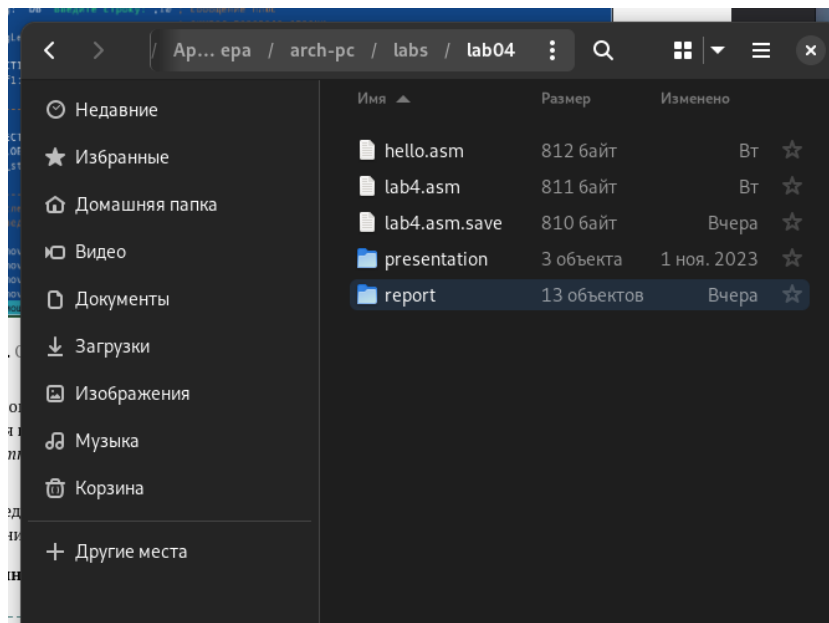


Рис. 3.8: файлы hello.asm и lab4.asm

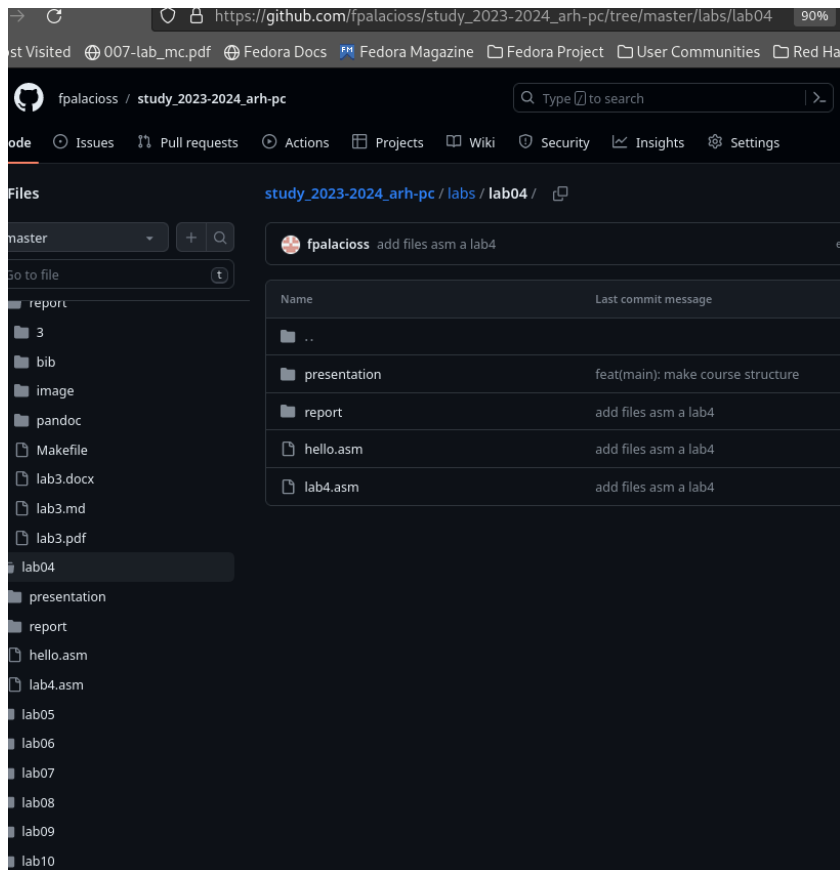


Рис. 3.9: Github



## 4 Выводы

Были освоены процедуры компиляции и сборки программ, написанных на ассемблере NASM

## **Список литературы**